

CUNY High Performance Computing Center

User Manual

V.2015.07

Table of Contents

1	Overview of the CUNY HPC Center resources.....	2
1.1	Organization of systems and data storage (architecture).....	2
1.2	HPC systems	3
1.2.1	Cluster/distributed memory computers	4
1.2.2	Symmetric Multi-processor systems (SMP)	5
2	Important administrative information.....	5
2.1	How to get an account	5
2.1.1	User accounts	5
2.1.2	Project accounts	6
2.2	Message of the day (MOTD)	6
2.3	Required citations.....	6
2.4	Reporting requirements.....	6
2.5	CUNY HPC systems' naming conventions	7
2.6	Funding	8
3	Running jobs	8
3.1	Input file on /scratch.....	8
3.2	Running jobs on "cluster" HPC systems.....	9
3.2.1	Serial (Scalar) Jobs.....	12
3.2.2	OpenMP or threaded Jobs	13
3.2.3	MPI Distributed Memory Parallel Jobs	13
3.2.4	GPU-Accelerated Data Parallel Jobs	14
3.2.5	Submitting jobs for execution.....	15
3.3	Running jobs on shared memory systems.....	15
3.4	Saving output files and clean-up.....	15
4	The Data Storage and Management System.....	15
4.1	"Project" directories.....	16
4.2	/global/u	17
4.3	/scratch.....	17
4.4	Data retention and account expiration policy	18
4.5	DSMS Technical Summary.....	19
4.6	Good data handling practices.....	20
4.6.1	DSMS, i.e., /global/u and SR1.....	20
4.6.2	/scratch	20
5	File Transfers	21
6	Modules and available third party software.....	22
6.1	Modules	22
6.1.1	Modules for the novice user.....	22
6.1.2	Modules for the advanced user.....	23
6.2	Application software.....	25
7	Training.....	26
8	Workshops	26
9	Appendix I: Job submit scripts for applications	26

1 Overview of the CUNY HPC Center resources

1.1 Organization of systems and data storage (architecture)

The CUNY High-performance Computing (HPC) Center is "data and storage centric", that is, it operates under the philosophy that "compute systems" are transient and will be periodically replaced, but research data is more permanent. Consequently, the environment is architected with a central file system, called the Data Storage and Management System (**DSMS**) with HPC systems attached to it. Figure 1 is a schematic of the environment. Here, the storage facilities, i.e., the **DSMS**, is in the center surrounded by the HPC systems. A HPC system can be added or removed from the system without affecting user data.

Access to all the HPC systems is through a "gateway system" called "chizen.csi.cuny.edu". This means that you must first sign into "**CHIZEN**" using ssh and then onto the HPC system you wish to use.

User home directories, user data, and project data are kept on the **DSMS**.

Each HPC system has local "/scratch" disk space. /scratch space is workspace used by the computer when running jobs. Input data required for a job, temporary data or intermediary files, and output files created by a job can temporarily reside on /scratch, but have no permanence there.

When a user applies for and is granted an account, they are assigned a **<userid>** on the HPC systems and the following disk space:

- /scratch/**<userid>** – this is temporary workspace on the HPC system
- /global/u/**<userid>** – space for "home directory", i.e., storage space on the **DSMS** for program, scripts, and data
- In some instances a user will also have use of disk space on the **DSMS** in /cunyZone/home/**<projectid>**.

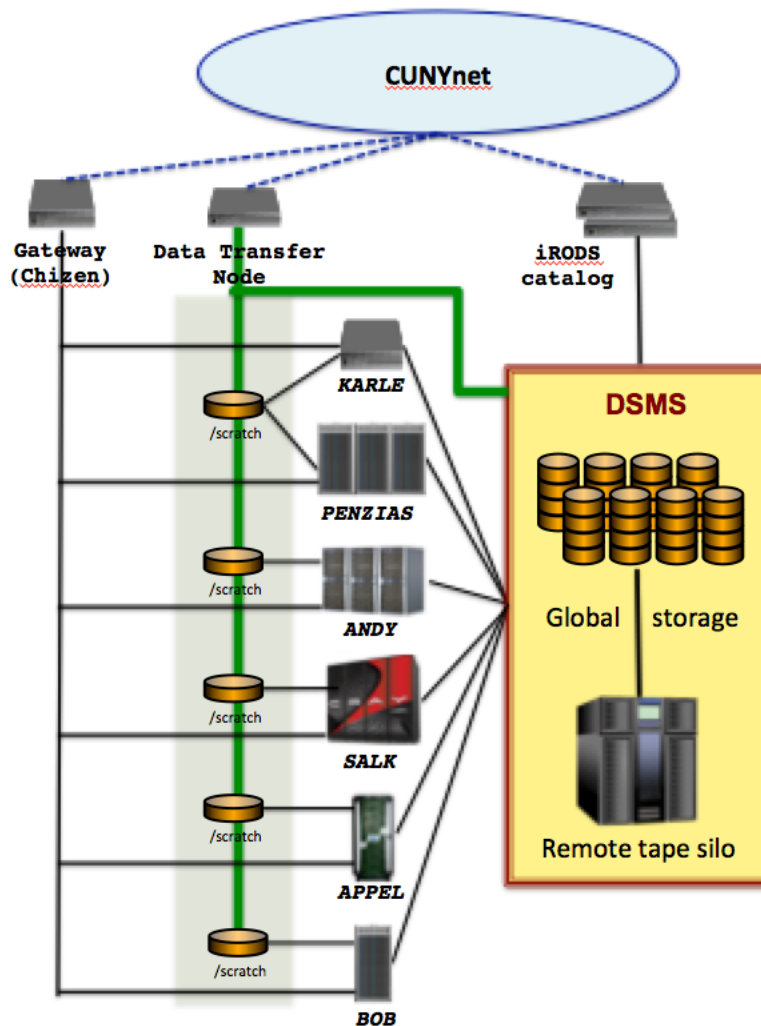


Figure 1. The **DSMS** is the global file system for the HPC Center. Chizen provides user access to any of the HPC systems. The Data Transfer Node allows users to transfer large files from remote sites directly to `/global/u/<userid>` or to `/scratch` on any of the HPC systems on which they have a `/scratch/<userid>`.

1.2 HPC systems

The HPC Center operates two different types of HPC systems: distributed memory (also referred to as “cluster”) computers and symmetric multiprocessor (also referred to as “shared-memory) computers (SMP). These systems are very different in architecture, programming models, and in the way they are used. A brief description of the differences between “cluster” and SMP computers is provided in the sections below. Table 1 provides a quick summary of the attributes of each of the systems available at the HPC Center.

Table 1. Production HPC systems							
System	Type	Job Mix	Nodes	Cores /node	Memory /node	Memory /core	Chip type
ANDY	Cluster	64 or fewer cores	93	8	24 Gbytes	3 Gbytes	Nehalem (2.93 GHz)
APPEL	SMP		1	384	12 Tbytes	NA	Ivy Bridge (3.0 GHz)
BOB	Cluster	Gaussian	28	8	16 Gbytes	2 Gbytes	AMD Barcelona
KARLE	SMP	Interactive and batch for some applications	1	24	96 Gbytes	NA	Penryn (2.4 GHz)
PENZIAs	Cluster	128 or fewer cores	60	12	48 Gbytes	4 Gbytes	Sandy Bridge (2.2 GHz)
		4 cores/GPU	60	4 cores and 2 GPUs	16 Gbytes		NVIDIA K20 GPUs
SALK	Cluster	1024 or fewer cores	176	16	32 Gbytes	2 Gbytes	AMD Magny-cour (2.3 GHz)

1.2.1 Cluster/distributed memory computers

Today's typical HPC "cluster" or distributed memory systems such as **ANDY**, **BOB**, **PENZIAs**, and **SALK** consists of many "compute-nodes", a "service-node(s)".

- The "compute-nodes" are where jobs are executed. Cluster computers can have hundreds, thousands, or tens of thousands compute nodes. Each node has a limited amount of memory and the memory of each node is disjoint from the memory of all the other nodes in the system. Compute nodes are connected to each other via (generally) very a fast interconnect, which allows processes run on different nodes to communicate, by sending messages across the interconnect using special programming models such as the Message Passing Interface (MPI) Library model
- The "service-node(s)" manages the operation of the HPC system. For example, a software package, called "PBSpro" runs on a service-node and is the job queuing and management system. PBSpro manages the processing of user jobs and the allocation of the system resources.
- The "login-node" is your interface to the system and is a type of service-node.

These systems can run single processor serial jobs or multiple processor parallel jobs. Parallel jobs generally use MPI.

1.2.2 Symmetric Multi-processor systems (SMP)

An SMP system can be viewed as a single large compute node with many processors that have access to a single, large memory space. SMPs are designed to support jobs that use multiple processors and large amounts of shared memory using either "threads" or openSHMEM.

"**KARLE**" and "**APPEL**" are SMPs.

This section is in development.

2 Important administrative information

2.1 How to get an account

2.1.1 User accounts

CUNY faculty, research staff, collaborators at other universities, and public and private sector partners, and currently enrolled CUNY students (who MUST have a faculty sponsor) may receive authorization to use the CUNY HPC Center systems. Applications for accounts are accepted at any time, but accounts expire on 30 September and must be renewed before then.

A user account is issued to an individual user. Accounts are not to be shared. Users are responsible for protecting their passwords. Passwords are not to be shared.

The online user account application form can be found at:

<http://www.csi.cuny.edu/cunyhpc/application.php3>

Complete all parts including information on publications, funded projects, and resources required. By applying for and obtaining an account, the user agrees to the Center's [Acceptable Use Policy](#) and the [User Account and Password Policy](#).

Each research user account, upon creation, is provided a 50 GB home directory mounted as /global/u/<userid>. If required, a user may request an increase in the size of their home directory. The Center will endeavor to satisfy reasonable requests.

Student class accounts are provided a 10 GB home directory. Please note that class accounts and data will be deleted 30 days after the semester ends (unless otherwise agreed upon). Students are responsible for backing up their own data prior to the end of the semester.

When a user account is established, only the user has read/write to his files. The user can change his UNIX permissions to allow others in his group to read/write to his file.

Please be sure to notify the HPC Center if user accounts need to be removed or added to a specific group.

2.1.2 Project accounts

Users and/or groups requiring additional disk storage and/or iRods accounts are required to fill out a Project Application Form (PAF). This form is to be filled out by the principal investigator (PI) of the project, and will provide project details including but not limited to: project and grant information, group members required access shared project files, and project needs (disk space, software, etc.).

All members of the group will need to fill out a User Account Form (UAF) before accounts and access can be granted to them. Supervisors and/or their designated project managers will be responsible for providing access and/or limitations to their assigned group members. [Details on this process are described in the Projects section.]

The Project Application Form can be found at the following link:
<http://www.csi.cuny.edu/cunyhpc/Accounts.html>

2.2 Message of the day (MOTD)

Users are encouraged to read the "Message of the day" (MOTD), which is displayed to the user upon logging onto a system. The MOTD provides information on scheduled maintenance time when systems will be unavailable and/or important changes in the environment that are of import to the user community. The MOTD is the HPC Center's only efficient mechanism for communicating to the broader user community as bulk email messages are often blocked by CUNY SPAM filters.

2.3 Required citations

The CUNY HPC Center appreciates the support it has received from the National Science Foundation (NSF). It is the policy of NSF that researchers who are funded by NSF or who make use of facilities funded by NSF acknowledge the contribution of NSF by including the following citation in their papers and presentations:

"This research was supported, in part, under National Science Foundation Grants CNS-0958379, CNS-0855217, ACI-1126113 and the City University of New York High Performance Computing Center at the College of Staten Island."

The HPC Center, therefore, requests its users to follow this procedure as it helps the Center demonstrate that NSF's investments aided the research and educational missions of the University.

2.4 Reporting requirements

The Center reports on its support of the research and educational community to both NSF and CUNY on an annual basis. Citations are an important factor that is included in these reports. Therefore, the Center requests all users to send copies of research papers developed, in part, using the HPC Center resources to hpchelp@csi.cuny.edu. This

also helps the Center to keep abreast of user research requirement directions and needs.

2.5 CUNY HPC systems' naming conventions

The Center names its systems after noteworthy CUNY alumni. There are many good reasons for this:

- It honors the accomplishments of these alumni.
- It informs or reminds students of the accomplishments of former CUNY students and, hopefully, inspires them.
- It heightens public awareness of the contributions of these alumni and of the role played by CUNY.

The current systems at the HPC Center are named after Kenneth Appel, Bruce Chizen, Andy Grove, Jonas Salk, Robert Kahn, and Arno Penzias. More information on each of these persons and systems follows:

"ANDY" is named in honor of Dr. Andrew S. Grove, a City College alumnus and one of the founders of the Intel Corporation. It is an SGI cluster with 744 processor cores. **"ANDY"** is for jobs using 64 cores or fewer and for Gaussian jobs.

"APPEL" is named in honor of Dr. Kenneth Appel (pronounced ah-PEL), an alumnus of Queens College. Appel, along with Wolfgang Haken, used computers to assist in proving the 4-color theorem. Appel said, *"Most mathematicians, even as late as the 1970s, had no real interest in learning about computers. It was almost as if those of us who enjoyed playing with computers were doing something nonmathematical or suspect."* **"APPEL"** is a SGI UV300 with 384 cores and 12 terabytes of shared memory—a system nicely configured to solve problems in computational group theory—and group theory was Appel's area of research.

"BOB" is named in honor of Dr. Robert E. Kahn, an alumnus of the City College, who, along with Vinton G. Cerf, invented the TCP/IP protocol. **"BOB"** is a Dell cluster with 232 processor cores. **"BOB"** supports users running Gaussian09; no other applications are supported on **"BOB"**.

"CHIZEN" is named in honor of Bruce Chizen, former CEO of Adobe, and a Brooklyn College alumnus. **"CHIZEN"** is the system that is used as a gateway to the above HPC systems. It is not used for computations.

"KARLE" is named in honor of Dr. Jerome Karle, an alumnus of the City College of New York who was awarded the Nobel Prize in Chemistry in 1985. **"KARLE"** is a Dell shared memory system with 24 processor cores. **"KARLE"** is used for serial jobs, Matlab, SAS, parallel Mathematica, and certain ARCview jobs. It is the only system that supports running interactive jobs relying on graphical user interface.

"PENZIAS" is named in honor of Dr. Arno Penzias, a Nobel Laureate in Physics, and a City College alumnus. **"PENZIAS"** is a cluster with 1,152

Intel Sandy Bridge cores each with 4 Gbytes of memory. It is used for applications requiring up to 128 cores. It also supports 136 NVIDIA Kepler K20 accelerators.

"SALK" is named in honor of Dr. Jonas Salk, the developer of the first polio vaccine, and a City College alumnus. It is a Cray XE6m with a total of 2,816 processor cores. **"SALK"** is reserved for large parallel jobs, particularly those requiring more than 64 cores. Emphasis is on applications in the environmental sciences and astrophysics.

2.6 Funding

The systems at Center were funded as follows:

DSMS	NSF Grant ACI-1126113
ANDY	NSF Grant CNS-0855217 and the New York City Council through the efforts of Borough President James Oddo
APPEL	New York State Regional Economic Development Grant through the efforts of State Senator Diane Savino
PENZIAS	The Office of the CUNY Chief Information Officer
SALK	NSF Grant CNS-0958379 and a New York State Regional Economic Development Grant

3 Running jobs

In this section, we discuss the process for running jobs on a HPC system. Typically the process involves the following:

- Having input files within your /scratch/<userid> directory on the HPC system you wish to use.
- Creating a job submit script that identifies the input files, the application program you wish to use, the compute resources needed to execute the job, and information on where you wish to write your output files.
- Submitting the job script.
- Saving output to the **DSMS**.

These steps are explained below.

3.1 Input file on /scratch

The general case is that you will have input files that have data on which you wish to operate. To compute or work on these files, they must be stored within the /scratch/<userid> directory of the HPC system you wish to use. These files can come from any of the following sources:

- You can create them using a text editor.
- You can copy them from your directory in the **DSMS**.
- You can copy input files from other places (such as your local

computer, the web, etc...)

A few examples follow:

If the files are in /global/u

```
cd /scratch/<userid>
mkdir <job_name> && cd <job_name>
cp /global/u/<userid>/<myTask>/a.out ./
cp /global/u/<userid>/<myTask>/<mydatafile> ./
```

If the files are in SR (cunyZone):

```
cd /scratch/<userid>
mkdir <job_name> && cd <job_name>
iget <myTask>/a.out ./
iget <myTask>/<mydatafile> ./
```

3.2 Running jobs on "cluster" HPC systems

To be able to schedule your job for execution and to actually run your job on one or more compute nodes, PBSpro needs to be instructed about your job's parameters. These instructions are typically stored in a "job submit script". In this section, we describe the information that needs to be included in a job submit script. The submit script typically includes

- job name
- queue name
- what compute resources (number of nodes, number of cores and the amount of memory, the amount of *local* scratch disk storage (*applies to Andy, Bob, and Penzias*), and the number of GPUs) or other resources a job will need
- packing option
- actual commands that need to be executed (binary that needs to be run, input/output redirection, etc.).

Documentation:

The PBSpro reference guide can be found at

<http://resources.altair.com/pbs/documentation/support/PBSProUserGuide12.1.pdf>

A pro forma job submit script is provided below.

```
#!/bin/bash
#PBS -q <queue_name>
#PBS -N <job_name>
#PBS -l select=<chunks>;ncpus=<cpus>
#PBS -l mem=<????>mb
#PBS -l place=<placement>
#PBS -V

# change to the working directory
cd $PBS_O_WORKDIR

echo ">>>> Begin <job_name>"

# actual binary (with IO redirections) and required input
# parameters is called in the next line
mpirun -np <chunks * ncpus> -machinefile $PBS_NODEFILE <Program Name>
<input_text_file> > <output_file_name> 2>&1

echo ">>>> Begin <job_name> Run ..."
```

Note: The "#PBS" string must precede every PBS parameter.

"#" symbol in the beginning of any other line designates a comment line which is ignored by PBSpro.

Explanation of PBSpro attributes and parameters:

-q <queue_name> The three available queues are "production", "development", and "interactive".

- "production" is the normal queue for processing your work.
- "development" is used when you are testing an application. Jobs submitted to this queue can not request more than 8 cores or use more than 1 hour of total CPU time. If the job exceeds these parameters, it will be automatically killed. "Development" queue has higher priority and thus jobs in this queue have shorter wait time.
- "interactive" is used for quick interactive tests. Jobs submitted into this queue run in an interactive terminal session on one of the compute nodes. They can not use more than 4 cores or use more than a total of 15 minutes of compute time.

-N <job_name> The user must assign a name to each job they run. Names can be up to 15 alphanumeric characters in length.

-l select=<chunks>; A "chunk" is a collection of resources (cores, memory, disk space etc...).

-l ncpus=<cpus> The number of cpus (or cores) that the user wants to use per chunk.

- Note: PBSpro refers to "cores" as "cpus"; historically, this was true, but processing units on a node are more typically referred

to as "cores".

-l mem=<mem>mb This parameter is optional. It specifies how much memory is needed per chunk. If not included, PBSpro assumes a default memory size on a per cpu (core) basis.

-l ngpus=<gpus> The number of graphics processing units that the user wants to use on a node (This parameter is only available on **PENZIAS**).

-l place=<placement> This parameter tells PBSpro how to distribute requested chunks of resources across nodes. "placement" can take one of three values: free, scatter or pack.

- If you select "free", PBSpro will place your job "chunks" on any nodes that have the required number of available resources.
- If you select "scatter", PBSpro will schedule your job so that only one chunk is taken from any virtual compute node.
- If you select "pack", PBSpro will only schedule your job to take all the requested chunks from one node (and if no such node is available job will be queued up)

Special note for MPI users

How the "place", "scatter", "select", and "ncpus" parameters are defined can significantly affect the run time of a job. For example, assume you need to run a job that requires 64 cores. This can be scheduled in a number of different ways. For example,

```
#PBS -l place=free
#PBS -l select=8:ncpus=8
```

will freely place the 8 job chunks on any nodes that have 8 cpus available. While this may minimize communications overhead in your MPI job, PBS will not schedule this job until 8 nodes each with 8 free cpus becomes available. Consequently, the job may wait longer in the input queue before going into execution.

```
#PBS -l place=free
#PBS -l select=32:ncpus=2
```

will freely place 32 chunks of 2 cores each. There will possibly be some nodes with 4 free chunks (and 8 cores) and there may be nodes with only 1 free chunk (and 2 cores). In this case, the job ends up being more sparsely distributed across the system and hence the total averaged latency may be larger then in case with select=8:ncpus=8

In this example, however, it will be much easier for PBS to run the job – it won't need to wait for 8 completely empty nodes. Therefore even though "select=32:ncpus=2" may probably execute slower, it has a higher chance to start faster and hence complete sooner.

If the following parameters are selected:

```
#PBS -l place=scatter
#PBS -l select=32:ncpus=2
```

PBS will distribute the chunks of 2 cpus across 32 nodes.

`mpirun -np <chunks * ncpus>`. This script line is only to be used for MPI jobs and defines the total number of cores required for the parallel MPI job.

The Table 2 below shows the maximum values of the various PBS parameters by system. Request only the resources you need as requesting maximal resources will delay your job.

Table 2. Maximum PBS settings by system

	np (1)	ncpus	ngpus	Mem/core (2)	Mem/ chunk (3)
ANDY	64	8	NA	2880	23040
BOB	1	8	NA	1920	15360
PENZIAS	128	12	2	3800	30400
SALK	768	4	NA	1920	7680

Notes:

NA = Resource Not Available on this system.

(1) Largest mpi job allowed on the system.

(2) Default memory size allocated/core is the shown maximum value.

(3) Requesting all of the memory on a node will result in the job staying in the input queue until a node becomes fully available; "select" must always be set to 1 chunk.

3.2.1 Serial (Scalar) Jobs

For serial jobs, "select=1" and "ncpus=1" should be used.

```
#!/bin/bash
#
# Typical job script to run a serial job in the production queue
#
#PBS -q production
#PBS -N <job_name>
#PBS -l select=1:ncpus=1
#PBS -l place=free
#PBS -V

# Change to working directory
```

```
cd $PBS_O_WORKDIR

# Run my serial job
</path/to/your_binary> > <my_output> 2>&1
```

3.2.2 OpenMP or threaded Jobs

OpenMP jobs can only run on a single virtual node. Therefore, for OpenMP jobs, "place=pack" and "select=1" should be used; "ncpus" should be set to [2, 3, 4,... n] where n must be less than or equal to the number of cores on a virtual compute node.

Typically, OpenMP jobs will use the **<mem>** parameter and may request up to all the available memory on a node.

```
#!/bin/bash
#
# Typical job script to run a 4-processor OpenMP job in 1 "chunk" in
# the production queue
#
#PBS -q production
#PBS -N <job_name>
#PBS -l select=1:ncpus=4:mem=<mem>mb
#PBS -l place=pack
#PBS -V

# Change to working directory
cd $PBS_O_WORKDIR

export OMP_NUM_THREADS=4
# Run my OpenMP job
</path/to/your_binary> > <my_output> 2>&1
```

Note for **SALK** (Cray XE) users: all jobs run on the Cray's compute nodes must be started with Cray's *aprun* command. In the above script, the last line will need to be modified to:

```
aprun -n 4 </path/to/your_binary> > <my_output> 2>&1
```

Option '-n 4' in the above line specifies number of cores that 'aprun' will use to start the job. It should not exceed the number of cores requested in the '-l select' statement.

3.2.3 MPI Distributed Memory Parallel Jobs

For an MPI job, "select=" and "ncpus=" can be one or more, with "np=" >/=1.

```
#!/bin/bash
#
# Typical job script to run a distributed memory MPI job in the
# production queue requesting 16 "chunks" each with one 1 cpu.
#
#PBS -q production
#PBS -N <job_name>
#PBS -l select=16:ncpus=1
#PBS -l place=free
#PBS -V

# Change to working directory
cd $PBS_O_WORKDIR

# Run my 16-core MPI job

mpirun -np 16 </path/to/your_binary> > <my_output> 2>&1
```

Note for **SALK** (Cray Xe6m) users: all jobs run on the Cray's compute nodes must be started with Cray's *aprun* command. In the above script, the last line will need to be modified to:

```
aprun -n 16 </path/to/your_binary> > <my_output> 2>&1
```

Option '-n 16' in the above line specifies number of cores that 'aprun' will use to start the job. It should not exceed number of cores requested in the '-l select' statement.

3.2.4 GPU-Accelerated Data Parallel Jobs

```
#!/bin/bash
#
# Typical job script to run a 1 CPU, 1 GPU batch job in the
# production queue
#
#PBS -q production
#PBS -N <job_name>
#PBS -l select=1:ncpus=1:ngpus=1
#PBS -l place=free
#PBS -V

# Find out which compute node the job is using
hostname

# Change to working directory
cd $PBS_O_WORKDIR

# Run my GPU job on a single node using 1 CPU and 1 GPU.
</path/to/your_binary> > <my_output> 2>&1
```

3.2.5 Submitting jobs for execution

NOTE: We do not allow users to run batch jobs on the login-node. It is acceptable to do short compiles on the login node, but all other jobs must be run by handing off the "job submit script" to PBSpro running on the head-node. PBSpro will then allocate resources on the compute-nodes for execution of the job.

The command to submit your "job submit script" (*<job.script>*) is:

```
qsub <job.script>
```

3.3 Running jobs on shared memory systems

This section is in development

3.4 Saving output files and clean-up

Normally you expect certain data in the output files as a result of a job. There are a number of things that you may want to do with these files:

- Check the content of these outputs and discard them. In such case you can simply delete all unwanted data with 'rm' command.
- Move output files to your local workstation. You can use 'scp' for small amounts of data and/or GlobusOnline for larger data transfers.
- You may also want to store the outputs at the HPCC resources. In this case you can either move your outputs to /global/u or to SR1 storage resource.

In all cases your /scratch/*<userid>* directory is expected to be empty. Output files stored inside /scratch/*<userid>* can be purged at any moment (except for files that are currently being used in active jobs) located under the /scratch/*<userid>*/*<job_name>* directory).

4 The Data Storage and Management System

The CUNY HPC Center has installed a new, centralized Data Storage and Management System (**DSMS**). The **DSMS** is designed to provide (i) Principal Investigators with the tools needed to manage Project data in accordance with funding agency requirements, (ii) provide superior performance in conjunction with use of the computational resources of the Center, (iii) provide standard UNIX files systems for home directories, and (iv) provide an extensible petabyte disk file system.

Key features of the **DSMS** system include:

- "Project" directories in an Integrated Rule-Oriented Data-management System (iRODS) managed resource. Project directories exist in a "virtual file space" called "cunyZone" which contains a resource called "Storage Resource 1 (SR1)". For the purpose of this document, we will use the terminology SR1 to describe "Project file space."
- "User" home directories in a standard Unix file system called /global/u.
- Enhanced parallel scratch space on the HPC systems.
- Capability to support a searchable iRODS catalog of "Project" metadata annotations.
- Principal Investigator administrative control over critical "Project" files on a project basis.
- Backups.

The **DSMS** is the HPC Center's primary file system and is accessible from all existing HPC systems, except for "**BOB**"¹. It will similarly be accessible from all future HPC systems.

DSMS features are explained below.

4.1 "Project" directories.

"Project" directories are managed through iRODS and accessible through iRODS commands, not standard UNIX commands. In iRODS terminology, a "collection" is the equivalent of "directory".

A "Project" is an activity that usually involves multiple users and/or many individual data files. A "Project" is normally led by a "Principal Investigator" (PI), who is a faculty member or a research scientist. The PI is the individual responsible to the University or a granting agency for the "Project". The PI has overall responsibility for "Project" data and "Project" data management. To establish a Project, the PI completes and submits the online "Project Application Form".

The PI appoints or removes individuals from access to his Project by instructing the HPC Center to add or delete a userid from the Project directory using the "Project Application Form".

The PI can be or may delegate data administrative responsibilities to an administrator who is a member of his Project team. The Project administrator can authorize additional administrators who are members of the Project team. Administrator privileges include granting read/write access to Project data files.

The administrator can write iRODS "rules" that can require that descriptive "metadata" be added to a searchable catalog. The catalog is maintained by iRODS. More information on iRODS is available at <http://irods.org/documentation/>

¹ At present, there are no plans to migrate "**BOB**" to the **DSMS**.

Each Project is assigned a unique "Project Identifier" or *PID*.

Each member of the Project team can read or write files to the Project directory using iRODS commands in accordance with the privileges provided by the Project administrator. On writing a file into SR1, Project data and the writer's identity information is automatically captured and stored as metadata along with the date the file was created on SR1.

Project files are backed up on a daily basis.

4.2 */global/u*

/global/u is a standard Unix file system that holds the home directories of individual users. When users request and are granted an allocation of HPC resources, they are assigned a *userid* and a 50 GB allocation of disk space for home directories on */global/u/<userid>*. These "home" directories are on the *DSMS*, not on the HPC systems, but can be accessed from any Center system. This does not apply to *BOB*, where home directories remain physically on *BOB*.

All home directories are backed up on daily basis.

4.3 */scratch*

Disk storage on the HPC systems is used only for "scratch" files. "scratch" files are temporary and are **not backed up**. */scratch* is used by jobs queued for or in execution. Output from jobs may temporarily be located in */scratch*.

In order to submit a job for execution, a user must "stage" or "mount" the files required by the job to */scratch* from */global/u* using UNIX commands and/or from "SR1" using iRODS commands.

Files in */scratch* on a system are **automatically purged** when (1) usage reaches 70% of available space, **or** (2) file residence on scratch exceeds two weeks, whichever occurs first. The amount of scratch space on the systems has been increased and is now as indicated below:

System	Available scratch space
ANDY	25 terabytes
PENZIAS	120 terabytes
SALK	125 terabytes
KARLE	Shares PENZIAS' scratch space

The user is responsible for taking the actions necessary for moving files to be kept from */scratch* to */global/u* or SR1.

Backups. /global/u user directories and SR1 Project files are backed up automatically to a remote tape silo system over a fiber optic network. Backups are performed daily.

If the user deletes a file from /global/u or SR1, it will remain on the tape silo system for 30 days, after which it will be deleted and cannot be recovered. If a user, within the 30 day window finds it necessary to recover a file, the user must expeditiously submit a request to hpchelp@csi.cuny.edu.

Less frequently accessed files are automatically transferred to the HPC Center robotic tape system, freeing up space in the disk storage pool and making it available for more actively used files. The selection criteria for the migration are age and size of a file. If a file is not accessed for 90 days, it may be moved to a tape in the tape library – in fact to two tapes, for backup. This is fully transparent to the user. When a file is needed, the system will copy the file back to the appropriate disk directory. No user action is required.

4.4 Data retention and account expiration policy

Project directories on SR1 are retained as long as the project is active. The HPC Center will coordinate with the Principal Investigator of the project before deleting a project directory. If the PI is no longer with CUNY, the HPC Center will coordinate with the PI's departmental chair or Research Dean, whichever is appropriate.

For user accounts, current user directories under /global/u are retained as long as the account is active. If a user account is inactive for one year, the HPC Center will attempt to contact the user and request that the data be removed from the system. If there is no response from the user within three months of the initial notice, or if the user cannot be reached, the user directory will be purged.

4.5 DSMS Technical Summary

File space	Purpose	Accessibility	Quota	Backup policy	Purge policy
Scratch: /scratch/ <userid> on PENZIAS, ANDY, SALK, BOB	High performance parallel scratch files systems. Work area for jobs, data sets, restart files, files to be pre-/post processed. Temporary space for data that will be removed within a short amount of time.	Not globally accessible. Separate /scratch/<userid> exists on each system. Visible on login and compute nodes of each system and on the data transfer nodes.	None	No Backup.	Files older than 2 weeks are automatically deleted OR when scratch file system reaches 70% utilization
Home: /global/u /<userid>	User home filespace. Essential data should be stored here, such as user's source code, documents, and data sets.	Globally accessible on the login and on the data transfer nodes through native GPFS or NFS mounts	Nominally 50 GB	Yes, backed up nightly to tape. If the active copy is deleted, the most recent backup is stored for 30 days.	Not purged
Project: /SR1/<PID>	Project space allocations	Accessible on the login and on the data transfer nodes. Accessible outside CUNY HPC Center through iRODS.	Allocated according to project needs.	Yes, backed up nightly to tape. If the active copy is deleted, the most recent backup is stored for 30 days and retrievable on request, but the iRODS metadata may be lost.	Not purged

- SR1 is tuned for high bandwidth, redundancy, and resilience. It is not optimal for handling large quantities of small files. If you need to archive more than a thousand of files on SR1, please create a single archive using "tar".

- A separate `/scratch/<userid>` exists on each system. On **PENZIAS**, **SALK**, **KARLE**, and **ANDY**, this is a Lustre parallel file system, on **BOB** it is NFS. These `/scratch` directories are visible on the login and compute nodes of the system only and on the data transfer nodes, but are not shared across HPC systems.
- `/scratch/<userid>` is used as a high performance parallel scratch filesystem, for example, temporary files (e.g. restart files) should be stored here.
- There are no quotas on `/scratch/<userid>`, however any files older than 30 days are automatically deleted. Also, a cleanup script is scheduled to run every two weeks or whenever the `/scratch` disk space utilization exceeds 70%. Dot-files are generally left intact from these cleanup jobs.
- `/scratch` space is available to all users. If the scratch space is exhausted, jobs will not be able to run. Purge any files in `/scratch/<userid>`, which are no longer needed, even before the automatic deletion kicks in.
- `/scratch/<userid>` directory may be empty when you login, you will need to copy any files required for submitting your jobs (submission scripts, data sets) from `/global/u` or from `SR1`. Once your jobs complete copy any files you need to keep back to `/global/u` or `SR1` and remove all files from `/scratch`.
- Do not use `/tmp` for storing temporary files. The file system where `/tmp` resides in memory is very small and slow. Files will be regularly deleted by automatic procedures.
- `/scratch/<userid>` is not backed up and there is no provision for retaining data stored in these directories.

4.6 Good data handling practices

4.6.1 *DSMS*, i.e., `/global/u` and `SR1`

- The **DSMS** is not an archive for non-HPC users. It is an archive for users who are processing data at the HPC Center. "Parking" files on the **DSMS** as a back-up to local data stores is prohibited.
- Do not store more than 1,000 files in a single directory. Store collections of small files into an archive (for example, tar). Note that for every file, a stub of about 4MB is kept on disk even if the rest of the file is migrated to tape, meaning that even migrated files take up some disk space. It also means that files smaller than the stub size are never migrated to tape because that would not make sense. Storing a large number of small files in a single directory degrades the file system performance.

4.6.2 `/scratch`

- Please regularly remove unwanted files and directories and avoid keeping duplicate copies in multiple locations. File transfer among the HPC Center systems is very fast. It is forbidden to use "touch jobs" to prevent the cleaning policy from automatically

deleting your files from the `/scratch` directories. Use `"tar -xmvf"`, not `"tar -xvf"` to unpack files. `"tar -xmvf"` updates the times stamp on the unpacked files. The `"tar -xvf"` command preserves the time stamp from the original file and not the time when the archive was unpacked. Consequently, the automatic deletion mechanism may remove files unpacked by `"tar -xvf"`, which are only a few days old.

5 File Transfers

The Center has deployed Data Transfer Nodes (DTN), which are servers dedicated to transfer data between remote sites and the **DSMS** (`/global/u/<userid>`), or between remote sites and the `/scratch/<userid>` on the designated HPC system on which the user has an account. DTNs are tuned to transfer data efficiently.

There are three methods of transferring data between the CUNY HPC systems and the rest of the world:

- **Globus Online:** The preferred method for large files, with extra features for parallel data streams, auto-tuning and auto-fault recovery. Globus online is to transfer files between systems—between the CUNY HPC Center resources and XSEDE facilities, or even users' desktops. A typical transfer rate ranges from 100 to 400 Mbps.
- **SCP/SFTP:** For smaller files (<1GB). Secure Copy (SCP) and Secure FTP (SFTP) can be used to securely transfer files between two hosts using the Secure Shell (SSH) protocol. A typical transfer rate ranges from 1 to 30 megabytes/second.
- **iRODS:** The data grid/data management tool provided by CUNY HPC Center for accessing the "SR1" resource. iRODS clients (`iput`, `iget`, `irsync`) provide a data transfer mechanism featuring bulk upload and parallel streams. Not all methods are offered for all file systems. Here is a summary on the available methods per file space:

File system	Available transfer methods	URL
User:	Globus Online	cunyhpc#cea
<code>/global/u/<userid></code>	SCP/SFTP	cea.csi.cuny.edu
Project: SR1/ <code><PID></code>	iRODS	irods.csi.cuny.edu
Scratch:	Globus Online	cunyhpc#cea
	SCP/SFTP	cea.csi.cuny.edu

/scratch/<userid> on PENZIAS		
/scratch/<userid> on ANDY		
/scratch/<userid> on KARLE		
/scratch/<userid> on SALK		
/scratch/<userid> on BOB		

6 Modules and available third party software

6.1 Modules

"Modules" makes it easier for users to run a standard or customized application and/or system environment.

"Modules" allows users to set environmental variables that are specific to their compilation, parallel programming, and/or application requirements. "Modules" makes it convenient to select different compiler, parallel programming, or applications versions. "Modules" can adjust the environment in an orderly way, altering or setting environmental variables as such PATH, MANPATH, LD_LIBRARY_PATH, etc.

Modules is available on **ANDY**, **KARLE**, **PENZIAS**, and **SALK**.

6.1.1 Modules for the novice user

The "Module" help facility provides basic descriptive information about the application version and purpose of the modules file.

To load a module enter:

```
> module load <module_name>
```

For documentation on "Modules":

```
> man module
```

For help enter:

```
> module help
```

To see a list of currently loaded "Modules" run:

```
> module list
```

To see a complete list of all modules available on the system run:

```
> module avail
```

6.1.2 Modules for the advanced user

A "Modules" example for advanced users who need to change their environment.

The HPC Center supports a number of different compilers, libraries, and utilities. In addition, at any given time different versions of the software may be installed. "Modules" is employed to define a default environment, which generally satisfies the needs of most users and eliminates the need for the user to create the environment. From time to time, a user may have a specific requirement that differs from the default environment.

In this example, the user wishes to use a version of the NETCDF library on the HPC Center's Cray Xe6 (**SALK**) that is compiled with the Portland Group, Inc. (PGI) compiler instead of the installed default version, which was compiled with the Cray compiler. The approach to do this is:

- Run "module list" to see what modules are loaded by default.
- Determine what modules should be unloaded.
- Determine what modules should be loaded.
- Add the needed modules, i.e., "module load"

The first step, see what modules are loaded, is shown below.

```
user@SALK:~> module list
```

```
Currently Loaded Modulefiles:
```

- 1) modules/3.2.6.6
- 2) nodestat/2.2-1.0400.31264.2.5.gem
- 3) sdb/1.0-1.0400.32124.7.19.gem
- 4) MySQL/5.0.64-1.0000.5053.22.1
- 5) lustre-cray_gem_s/1.8.6_2.6.32.45_0.3.2_1.0400.6453.5.1-1.0400.32127.1.90
- 6) udreg/2.3.1-1.0400.4264.3.1.gem
- 7) ugni/2.3-1.0400.4374.4.88.gem
- 8) gni-headers/2.1-1.0400.4351.3.1.gem
- 9) dmapp/3.2.1-1.0400.4255.2.159.gem
- 10) xpmem/0.1-2.0400.31280.3.1.gem
- 11) hss-llm/6.0.0
- 12) Base-opts/1.0.2-1.0400.31284.2.2.gem
- 13) xtpe-network-gemini
- 14) cce/8.0.7
- 15) acml/5.1.0
- 16) xt-libsci/11.1.00
- 17) pmi/3.0.0-1.0000.8661.28.2807.gem
- 18) rca/1.0.0-2.0400.31553.3.58.gem
- 19) xt-asyncpe/5.13
- 20) atp/1.5.1
- 21) PrgEnv-cray/4.0.46
- 22) xtpe-mc8
- 23) cray-mpich2/5.5.3
- 24) pbs/11.3.0.121723

From the list, we see that the Cray Programming Environment ("PrgEnv-cray/4.0.46") and the Cray Compiler environment are loaded ("cce/8.0.7") by default. To unload these Cray modules and load in the PGI equivalents we need to know the names of the PGI modules. The "module avail" command shows this.

```
user@SALK:~> module avail
```

```
.  
.   
.
```

We see that there are several versions of the PGI compilers and two versions of the PGI Programming Environment installed. For this example, we are interested in loading PGI's 12.10 release (not the

default, which is "pgi/12.6") and the most current release of the PGI programming environment ("PrgEnv-pgi/4.0.46"), which is the default.

The following module commands will unload the Cray defaults, load the PGI modules mentioned, and load version 4.2.0 of NETCDF compiled with the PGI compilers.

```
user@SALK:~> module unload PrgEnv-cray
user@SALK:~> module load PrgEnv-pgi
user@SALK:~> module unload pgi
user@SALK:~> module load pgi/12.10
user@SALK:~>
user@SALK:~> module load netcdf/4.2.0
user@SALK:~>
user@SALK:~> cc -V

/opt/cray/xt-asyncpe/5.13/bin/cc: INFO: Compiling with
CRAYPE_COMPILE_TARGET=native.

pgcc 12.10-0 64-bit target on x86-64 Linux
Copyright 1989-2000, The Portland Group, Inc. All Rights
Reserved.
Copyright 2000-2012, STMicroelectronics, Inc. All Rights
Reserved.
```

A few additional comments:

- The first three commands do not include version numbers and will therefore load or unload the current default versions.
- In the third line, we unload the default version of the PGI compiler (version 12.6), which is loaded with the rest of the PGI Programming Environment in the second line. We then load the non-default and more recent release from PGI, version 12.10 in the fourth line.
- Later, we load NETCDF version 4.2.0 which, because we have already loaded the PGI Programming Environment, will load the version of NETCDF 4.2.0 compiled with the PGI compilers.
- Finally, we check to see which compiler the Cray "cc" compiler wrapper actually invokes after this sequence of module commands by again entering "module list".

6.2 Application software

Table 2a is a list of application software organized alphabetically. The list also shows the system(s) on which the software is installed.

Table 2b is the list of the same software organized by application area. There is some "fuzziness" in this organization as some applications can be used in multiple discipline areas.

The on-line versions of these tables provide links to sample job scripts for each of the applications.

The scripts are also included in Appendix I, which is a separate, downloadable document.

<p>This section in in development</p>

7 Training

Trainings are held throughout the year and also offered based upon group and user requests. Upcoming training sessions can be viewed at: <http://www.csi.cuny.edu/cunyhpc/workshops.php>

If you would like to request a specific training session for your group, please contact us at HPCHelp@csi.cuny.edu. If you require one-on-one help, please stop by at the Graduate Center campus during our Helpdesk hours (<http://www.csi.cuny.edu/cunyhpc/workshops.php>) or email our [Helpline](#).

8 Workshops

Workshops are held throughout the year, upcoming training sessions can be viewed at: <http://www.csi.cuny.edu/cunyhpc/workshops.php>

9 Appendix I: Job submit scripts for applications

<p>This section in in development</p>
